**Team Members:**

Roshini Varada- 26

V H N V R S Sai Pavan -27

**Source\_code – express JS**

var createError = require('http-errors');

var express = require('express');

var path = require('path');

var favicon = require('serve-favicon');

var logger = require('morgan');

var mongoose = require('mongoose');

mongoose.connect('mongodb+srv://test:test@cluster0-1baqs.mongodb.net/test?retryWrites=true&w=majority')

  .then(() => console.log('connection successful'))

  .catch((err) => console.error(err));

  // var mongoose = require('mongodb').MongoClient;

  // const uri = "mongodb+srv://test:test@cluster0-1baqs.mongodb.net/test?retryWrites=true&w=majority";

  // const client = new MongoClient(uri, { useNewUrlParser: true });

  // client.connect(err => {

  //   const collection = client.db("test").collection("devices");

  //   // perform actions on the collection object

  //   client.close();

  // });

var apiRouter = require('./routes/book');

var app = express();

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({extended: false}));

app.use(express.static(path.join(\_\_dirname, 'dist/mean-angular6')));

app.use('/books', express.static(path.join(\_\_dirname, 'dist/mean-angular6')));

app.use('/book-details/:id', express.static(path.join(\_\_dirname, 'dist/mean-angular6')));

app.use('/book-create', express.static(path.join(\_\_dirname, 'dist/mean-angular6')));

app.use('/book-edit/:id', express.static(path.join(\_\_dirname, 'dist/mean-angular6')));

app.use('/api', apiRouter);

// catch 404 and forward to error handler

app.use(function (req, res, next) {

  next(createError(404));

});

// error handler

app.use(function (err, req, res, next) {

  // set locals, only providing error in development

  res.locals.message = err.message;

  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page

  res.status(err.status || 500);

  res.send(err.status);

});

module.exports = app;

**Source-code angular:**

**Navigation bar component:**

    <mat-toolbar color="primary">

      <span>Library management System</span>

    </mat-toolbar>

Navigation bar ts:

import { Component } from '@angular/core';

import { BreakpointObserver, Breakpoints, BreakpointState } from '@angular/cdk/layout';

import { Observable } from 'rxjs';

@Component({

  selector: 'my-nav',

  templateUrl: './my-nav.component.html',

  styleUrls: ['./my-nav.component.css']

})

export class MyNavComponent {

  isHandset: Observable<BreakpointState> = this.breakpointObserver.observe(Breakpoints.Handset);

  constructor(private breakpointObserver: BreakpointObserver) {}

}

Code for updating and deletion of record in mongo db

import {Injectable} from '@angular/core';

import {Observable, of, throwError} from 'rxjs';

import {HttpClient, HttpHeaders, HttpErrorResponse} from '@angular/common/http';

import {catchError, tap, map} from 'rxjs/operators';

const httpOptions = {

  headers: new HttpHeaders({'Content-Type': 'application/json'})

};

const apiUrl = '/api';

@Injectable({

  providedIn: 'root'

})

export class ApiService {

  constructor(private http: HttpClient) {

  }

  private handleError(error: HttpErrorResponse) {

    if (error.error instanceof ErrorEvent) {

      // A client-side or network error occurred. Handle it accordingly.

      console.error('An error occurred:', error.error.message);

    } else {

      // The backend returned an unsuccessful response code.

      // The response body may contain clues as to what went wrong,

      console.error(

        `Backend returned code ${error.status}, ` +

        `body was: ${error.error}`);

    }

    // return an observable with a user-facing error message

    return throwError('Something bad happened; please try again later.');

  }

  private extractData(res: Response) {

    let body = res;

    return body || {};

  }

  getBooks(): Observable<any> {

    return this.http.get(apiUrl, httpOptions).pipe(

      map(this.extractData),

      catchError(this.handleError));

  }

  getBook(id: string): Observable<any> {

    const url = `${apiUrl}/${id}`;

    return this.http.get(url, httpOptions).pipe(

      map(this.extractData),

      catchError(this.handleError));

  }

  postBook(data): Observable<any> {

    return this.http.post(apiUrl, data, httpOptions)

      .pipe(

        catchError(this.handleError)

      );

  }

  updateBook(id: string, data): Observable<any> {

    const url = `${apiUrl}/${id}`;

    return this.http.put(url, data, httpOptions)

      .pipe(

        catchError(this.handleError)

      );

  }

  deleteBook(id: string): Observable<{}> {

    const url = `${apiUrl}/${id}`;

    return this.http.delete(url, httpOptions)

      .pipe(

        catchError(this.handleError)

      );

  }

}